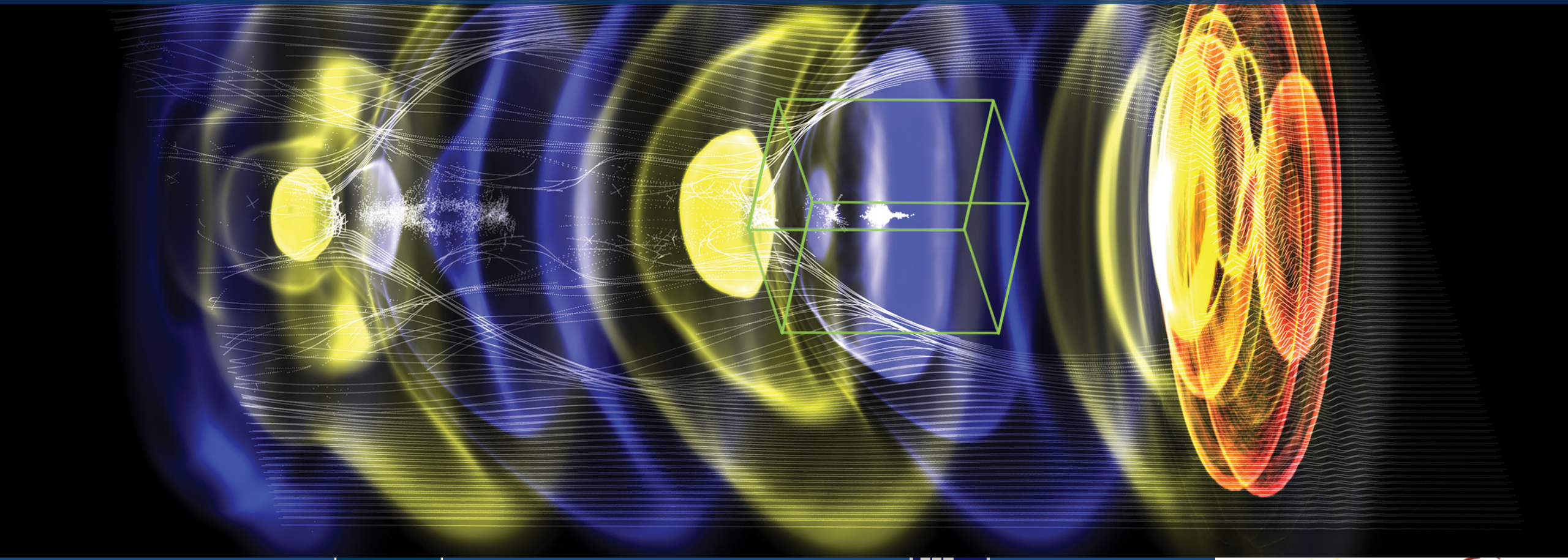
# WarpX: implementation and performance on GPU

R. Lehe, on behalf of the WarpX team



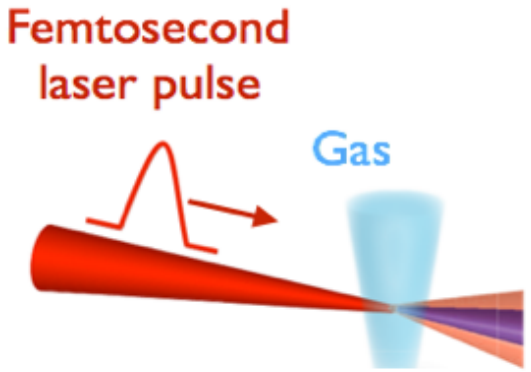
U.S. DEPARTMENT OF ENERGY | Office of Science
BERKELEY LAB
ACCELERATOR TECHNOLOGY & APPLIED PHYSICS DIVISION ATAP
SLAC
BLAST BERKELEY LAB ACCELERATOR SIMULATION TOOLKIT
ECP EXASCALE COMPUTING PROJECT

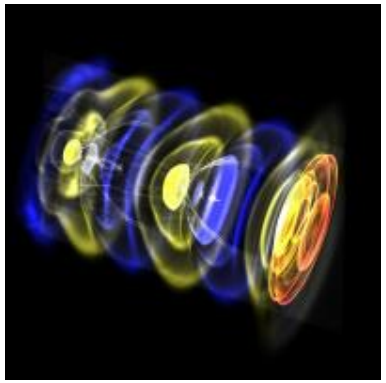
# Outline

- **Overview of WarpX and GPU implementation**
- **GPU performance, and lessons learned**

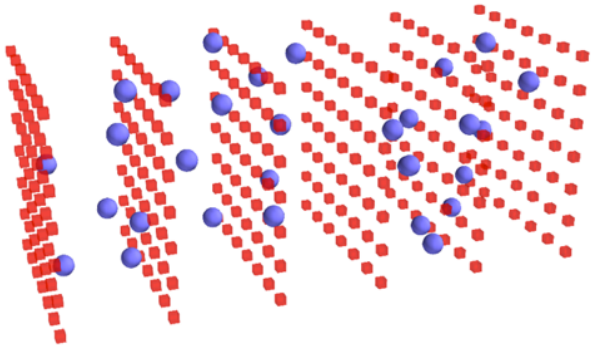
# Overview of WarpX

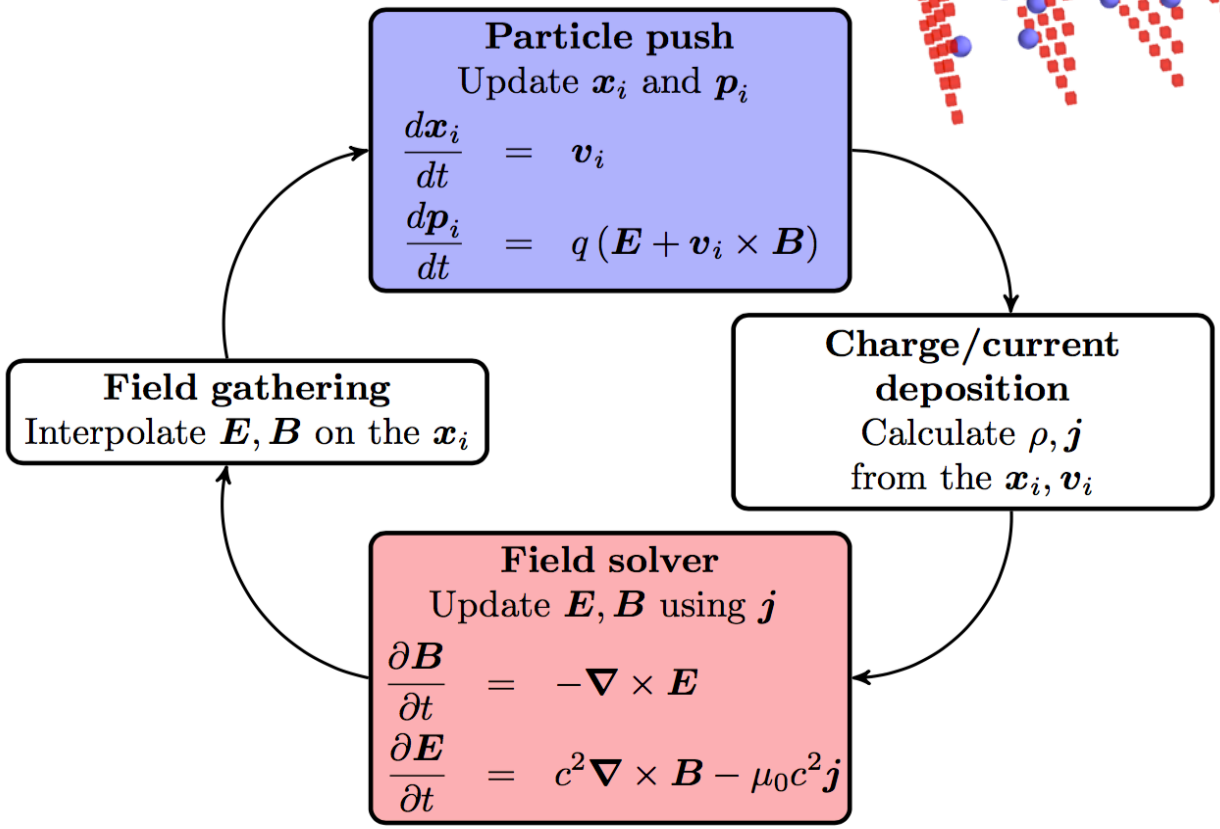
**Main purpose:**
model laser-plasma interactions



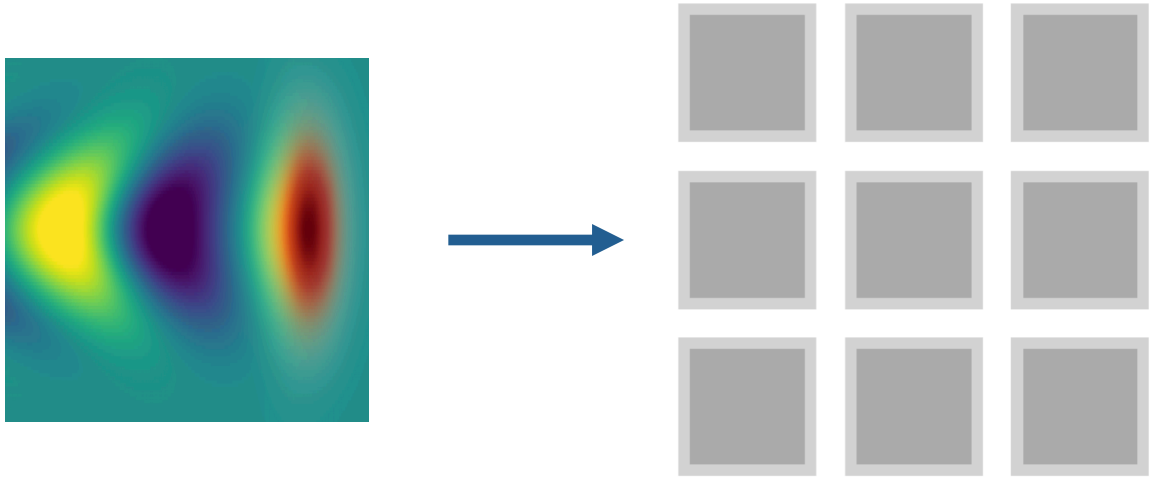
**Algorithm:**
electro-magnetic Particle-In-Cell





# WarpX: fundamental operations

**MPI parallelization:**
3D spatial domain decomposition



**“Compute” routines:**

- Particle push
- Current deposition (particle-to-grid)
- Field solver
- Field gathering (grid-to-particle)

**Communications routines: (between sub-domains)**

- Particle exchange
- Field exchange (guard cells)

# WarpX: Code structure

- **Memory allocation / management**
  - Handled by **AMReX**

- **"Compute" routines**
  - Custom code in **Fortran** and **C++**

    (for field gathering,
    current deposition, etc…)

- **MPI communications**
  - Call to **AMReX functions** (`FillBoundary`, Particle `Redistribute`)

# WarpX: Code structure on GPU

- **Memory allocation / management**
  - Handled by **AMReX**
  - Use **managed memory** (with pre-allocated memory pool)
  - User needs to make sure that simulation **fits** in GPU memory

- **"Compute" routines**
  - Custom code in **Fortran** and **C++**

    (for field gathering,
    current deposition, etc…)

**OpenACC (Fortran)**

```
!$acc parallel deviceptr(xp, zp, uxp, uzp, gaminv)
!$acc loop gang vector
DO ip=1, np
  xp(ip) = xp(ip) + uxp(ip)*gaminv(ip)*dt
  zp(ip) = zp(ip) + uzp(ip)*gaminv(ip)*dt
ENDDO
```

**AMReX GPU framework (C++)**

```
amrex::ParallelFor(bx,
    [=] AMREX_GPU_DEVICE (int i, int j, int k)
    {
        fab(i,j,k) += 1.;
    });
```

- **MPI communications**
  - Call to **AMReX functions** (`FillBoundary`, Particle `Redistribute`)
  - GPU-CPU copies (pinned memory) + CPU-CPU MPI exchanges

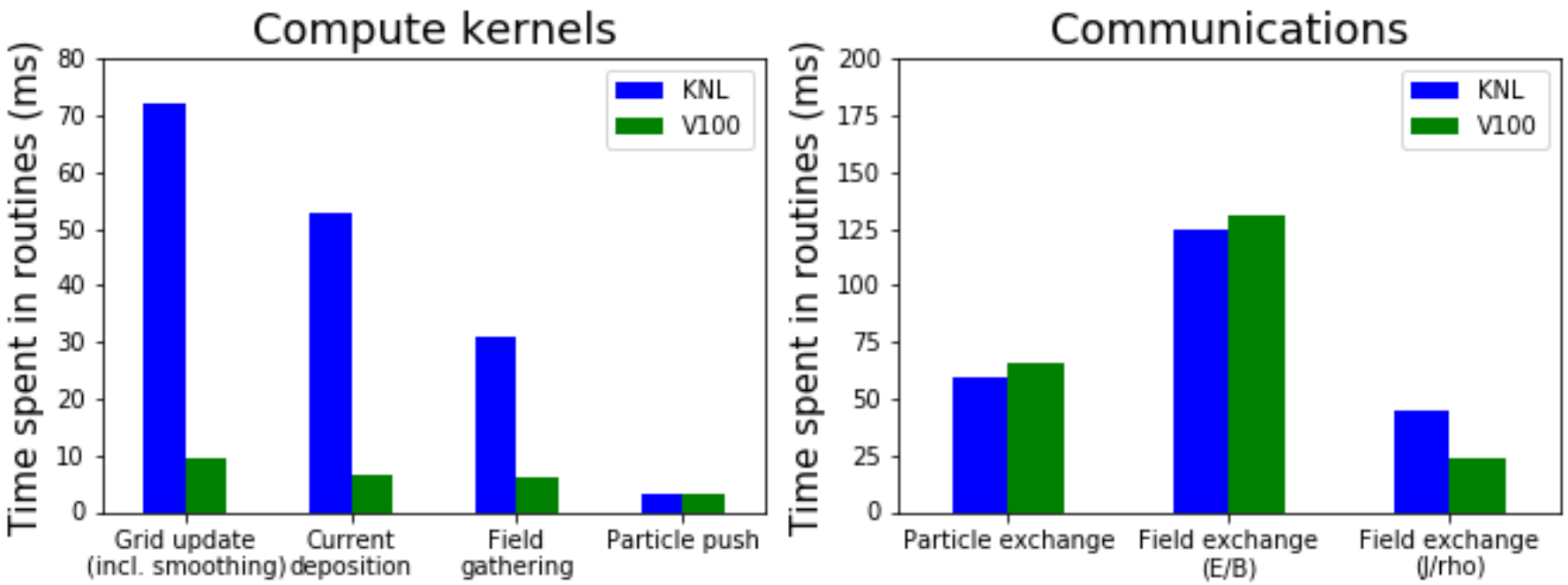
# Outline

- Overview of WarpX and GPU implementation
- GPU performance, and lessons learned

# Performance: CPU vs. GPU

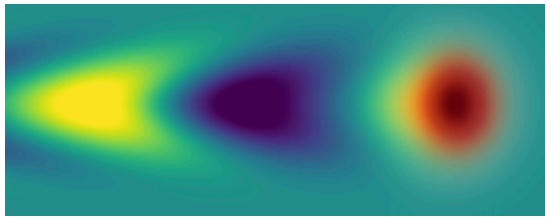
**Benchmark:** large-scale, production-type simulation on:
**900 KNL nodes (Cori)** vs. **900 V100 GPUs (Summit)**

**Lower is better**

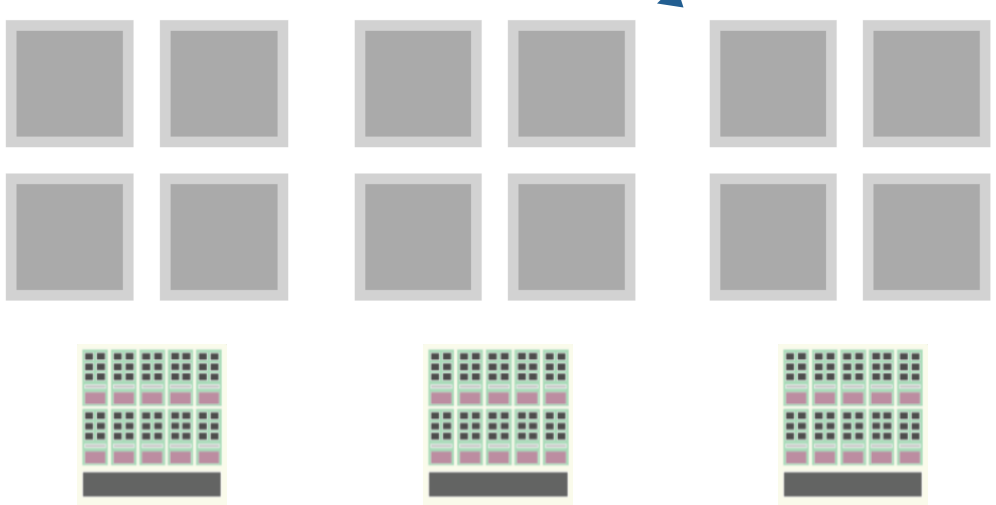


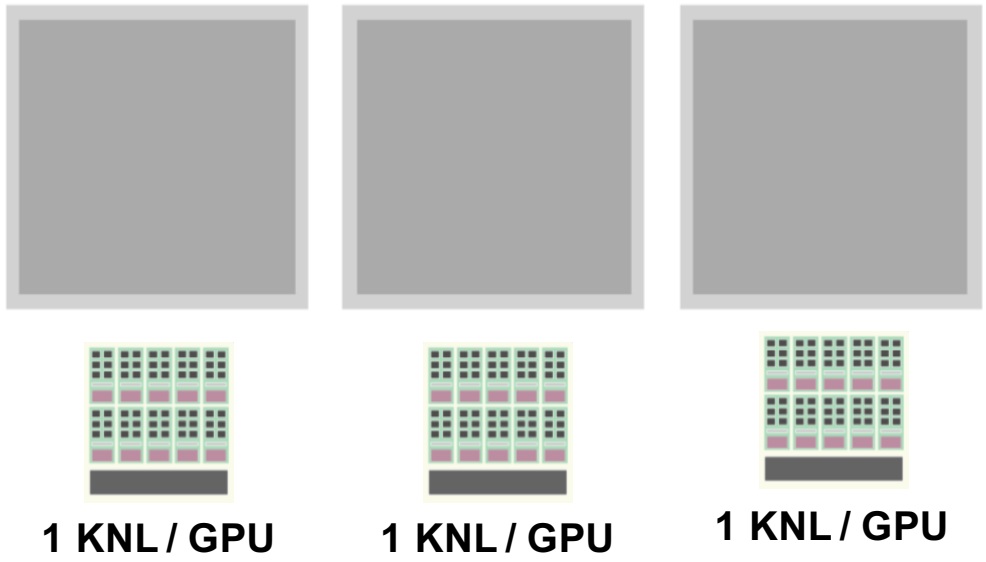

- Time spent in the **compute routines** themselves is **dramatically reduced!**
- Time spent in **communication routines** is **comparable.**
- Thus, the cost of MPI communications is **comparatively more important** on GPU.

# Reducing the cost of communications: Larger subdomains



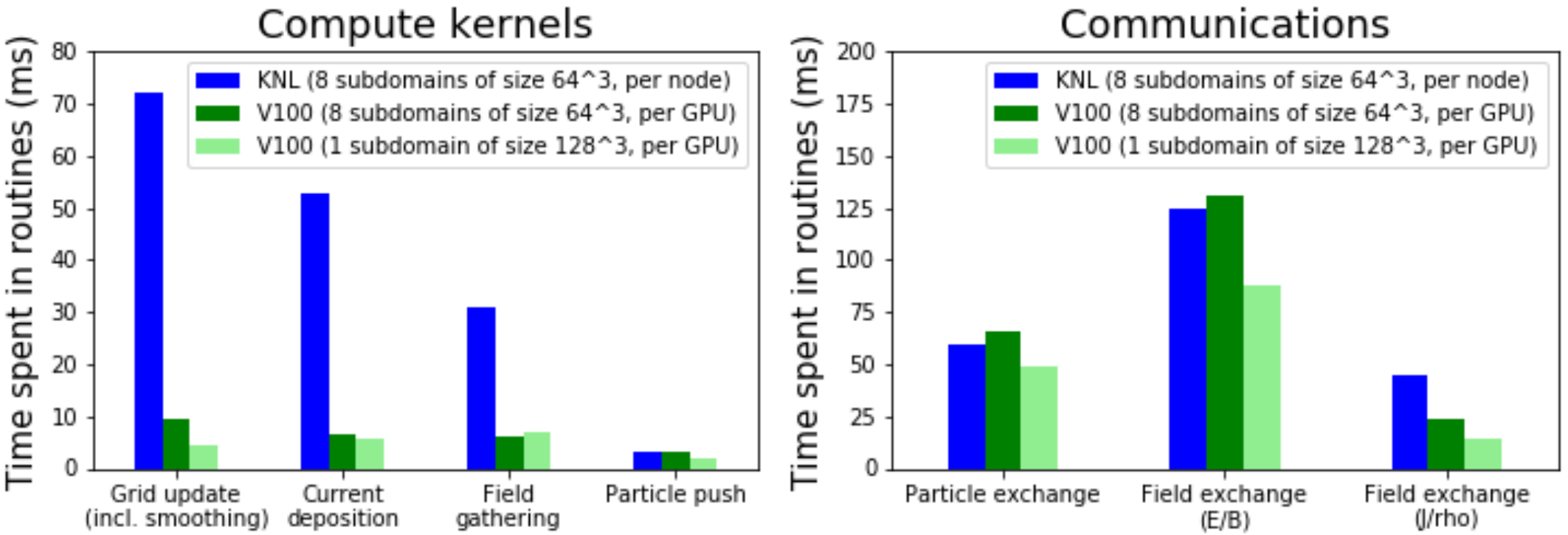
Possible domain decompositions




Using **1 sub-domain per KNL** is **not efficient** (imposes 1 MPI rank per KNL node).
But what about **1 sub-domain per GPU**?

# Reducing the cost of communications: Larger subdomains




Using **1 large sub-domain per GPU** instead of **several small sub-domains per GPU** reduces the overheads of communications.

# Summary and outlook

- **Status and performance**
  - WarpX has been ported to GPU
  - Performance of compute routines is considerably better than on KNL
  - Communications benefit from GPU's ability to use larger sub-domains

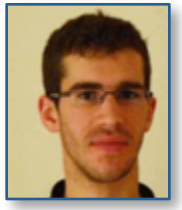
- **Near-future plans**
  - Move routines from Fortran/OpenACC to C++/AMReX GPU framework
  - Reduce cost of MPI communications (e.g. group more exchanges, cuda graph, etc..)
  - Optimization of individual routines

# Thanks!

**LBNL ATAP**

Jean-Luc Vay (PI)

Rémi Lehe



Jaehong Park

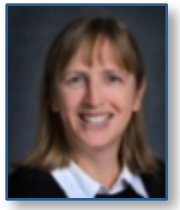
Olga Shapoval

Maxence Thevenet

Diana Amorim

**LBNL CRD**

Ann Almgren (coPI)



John Bell

Andrew Myers

Weiqun Zhang

Revathi Jambunathan

**SLAC**

Marc Hogan (coPI)

Lixin Ge

Cho Ng

**LLNL**

David Grote (coPI)

EXASCALE COMPUTING PROJECT